Supplementary Material:
Learning partial differential equations for biological transport models from noisy spatiotemporal data

John Lagergren, John T. Nardini, G. Michael Lavigne, Erica M. Rutter, Kevin B. Flores

S1 Error Model Selection

This section is based on [1, § 3.5], and we are concerned with the selection of an appropriate error model from a given data set, $\{U_{i,j}\}_{i=1,j=1}^{M,N}$. A common and flexible error model is given by the nonconstant variance error model

$$U_{i,j} = f(x_i, t_j|\theta_0) + f^{\gamma}(x_i, t_j|\theta_0)\epsilon_{i,j}, \tag{1}$$

in which $\epsilon_{i,j} \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$. This error model is flexible in that it can account for both constant variance error models (when $\gamma = 0$) and nonconstant variance error model (when $\gamma > 0$). In the latter case, we can quantify the extent of the variance's dependence on $f(x,t)$ with estimation of γ. Computing and plotting residuals is a useful way to estimate an appropriate value of γ from data.

From Equation (1), we observe that the *modified residuals*,

$$r_{i,j} = \frac{U_{i,j} - f(x_i, t_j|\theta)}{f^{\gamma}(x_i, t_j|\theta)}, i = 1, ..., M; j = 1, ..., N \tag{2}$$

should be MN realizations of an *i.i.d.* random variable when γ has been chosen correctly and $\theta \approx \theta_0$. One can thus choose the correct form of Equation 1 for a given data set and mathematical, $f(x,t;\theta)$, as follows: (i). Pick a value of γ, (ii). compute $\hat{\theta}$ by minimizing the generalized cost function, i.e., $\hat{\theta} = \arg\min_{\theta} \sum_{i=1,j=1}^{M,N} r_{i,j}^2$, and (iii). compute and plot each $r_{i,j}$. For the correct value of γ, the plotted modified residual computations will appear *i.i.d.* Note this method can be used for error models different from Equation (1). For example, Nardini and Bortz [3] used residual computations to demonstrate that a spatially-autocorrelated error model can account for numerical error arising from a PDE's discretization scheme during an inverse problem methodology.

As an example, we have plotted the modified results for the advection-diffusion dataset with $\sigma = 0.25$ in Figure S1 for $\gamma = 0, 0.5$, and 1.0. For each value of γ, we trained the ANN on the assumption that data was of the form given by Equation (1). For $\gamma = 0$ and 0.5, we observe that the residuals do not appear *i.i.d.*, as they fan out with increasing values of $u(x,t)$. At $\gamma = 1.0$, however, we see that the variance of the modified residuals appears constant, suggesting that $\gamma = 1.0$ is the appropriate value from the data.

S2 Comparing spline and bi-spline methods for denoising data

We compared the accuracy in learning the correct PDE when using 1-dimensional cubic CV splines versus cubic CV bi-splines for denoising data and approximating partial derivatives (Figures S2,S3,S4). We found that PDE-FIND with pruning always has a higher TPR value when using bi-spline computations as compared to 1-dimensional splines.

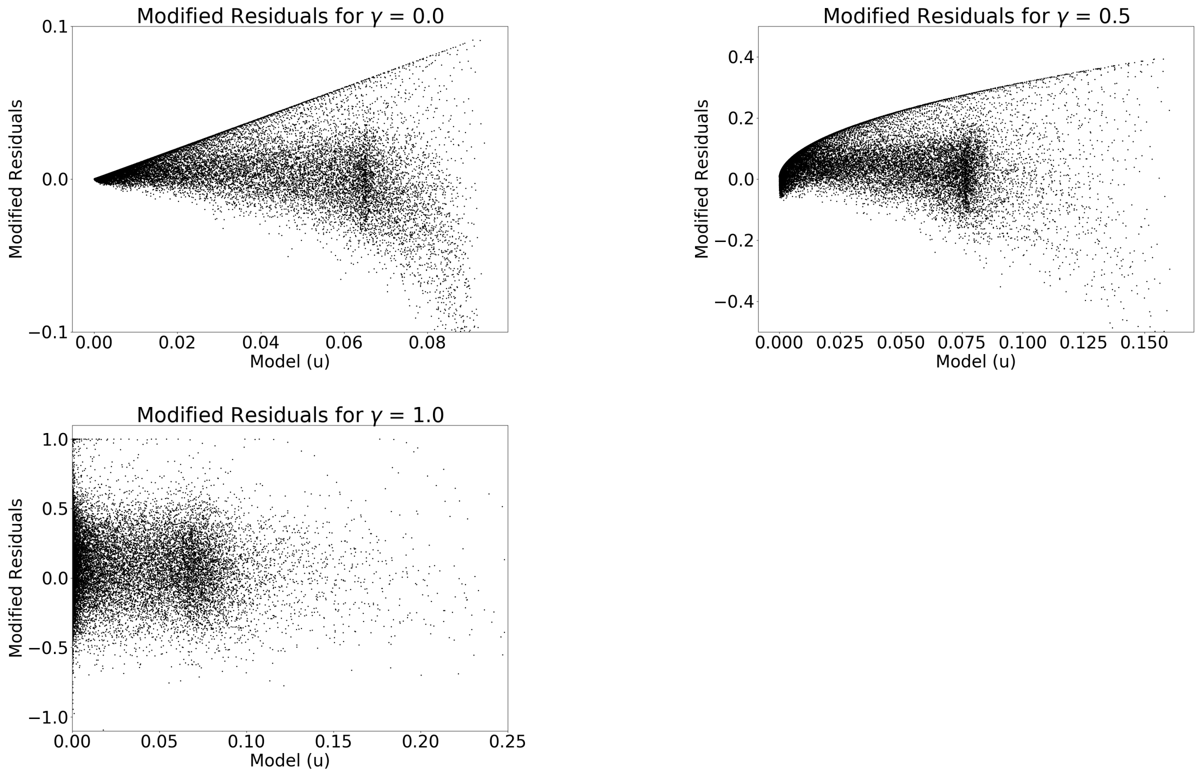

Figure S1: Modified Residual computations for various values of γ from the advection-diffusion data set with $\sigma = 0.25$. Top left: results for $\gamma = 0$, Top right: results for $\gamma = 0.5$, bottom left: results for $\gamma = 1.0$.

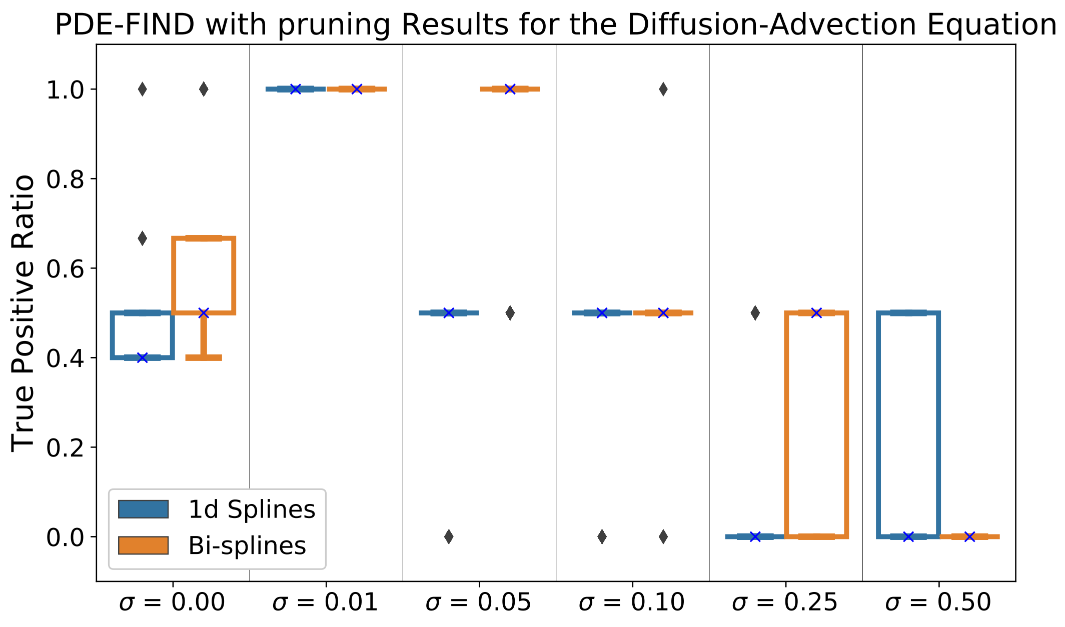

Figure S2: TPR values for the diffusion-advection equation when using 1-dimensional cubic splines versus cubic bi-splines for denoising data and approximating partial derivatives.

S3 Global Spline Calculations

We are concerned with approximating $\{U_i, j\}_{i=1,j=1}^{M,N}$ with a global spline representation of the form

$$S(x,t) = \sum_{k=1,l=1}^{K,L} c_{k,l} S_{k,l}(x,t), \tag{3}$$

where $S_{k,l}(x,t)$ are normalized bivariate cubic B-splines defined on the knot locations $(x,t)_{k,l} = \{\tilde{x}_k, ..., \tilde{x}_{k+4}\} \times \{\tilde{t}_l, ..., \tilde{t}_{l+4}\}$. In order to do so, we need to estimate a smoothing parameter, s, the knot locations, $\{(x,t)_{k,l}\}_{k=1,l=1}^{K,L}$, and the spline coefficients, $c = \{c_{k,l}\}_{k=1,l=1}^{K,L}$. To obtain these estimates, we split $\{U_{i,j}\}_{i=1,j=1}^{M,N}$ into a training and validation set (50%/50%). Recall that bivariate splines need to be fit to data on a rectangular grid domain, so we maintain this structure in the training and validation set by selecting all spatial points but only every other time point for the training data. The remaining points are placed in the validation data. While the (50%/50%) training and validation split here is not equivalent to the (90%/10%) split used to train the ANN, we found that implementing the global splines on a (90%/10%) took too much time for practical computation (training one data set took over a day on an Intel i7 6-core 3.5GHz desktop computer).

To find the optimal value of s, we set we begin with $s = MN + \sqrt{2MN}$, a previously-proposed upper bound on this smoothing parameter [2], and find the knot locations and coefficient values that minimize the GLS cost function $\mathcal{J}_S(\theta)$ (Equation (2.4) from in the main text) on the validation data. Further description of identification of spline locations of coefficients is given below. We then continue to divide s in half and re-compute $S(x,t)$ for the updated values of s until $\mathcal{J}_S(\theta)$ begins to increase on the validation data. We then compute $S(x,t)$ for a finer grid of s values around whichever value minimized $\mathcal{J}_S(\theta)$ and ultimately select whichever of these value minimize $\mathcal{J}_S(\theta)$

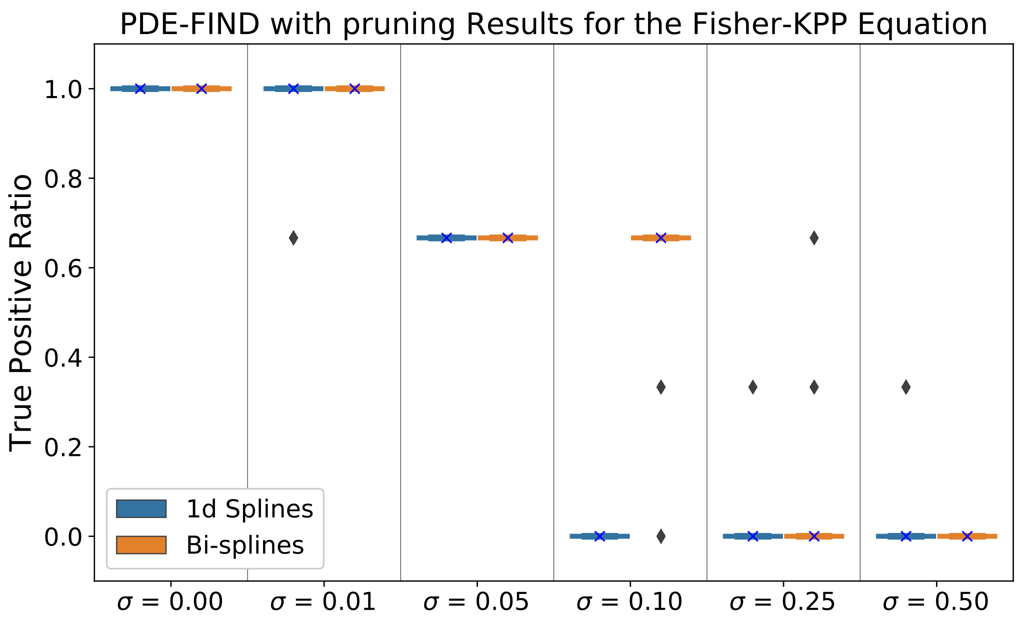

Figure S3: TPR values for the Fisher-KPP equation when using 1-dimensional cubic splines versus cubic bi-splines for denoising data and approximating partial derivatives.

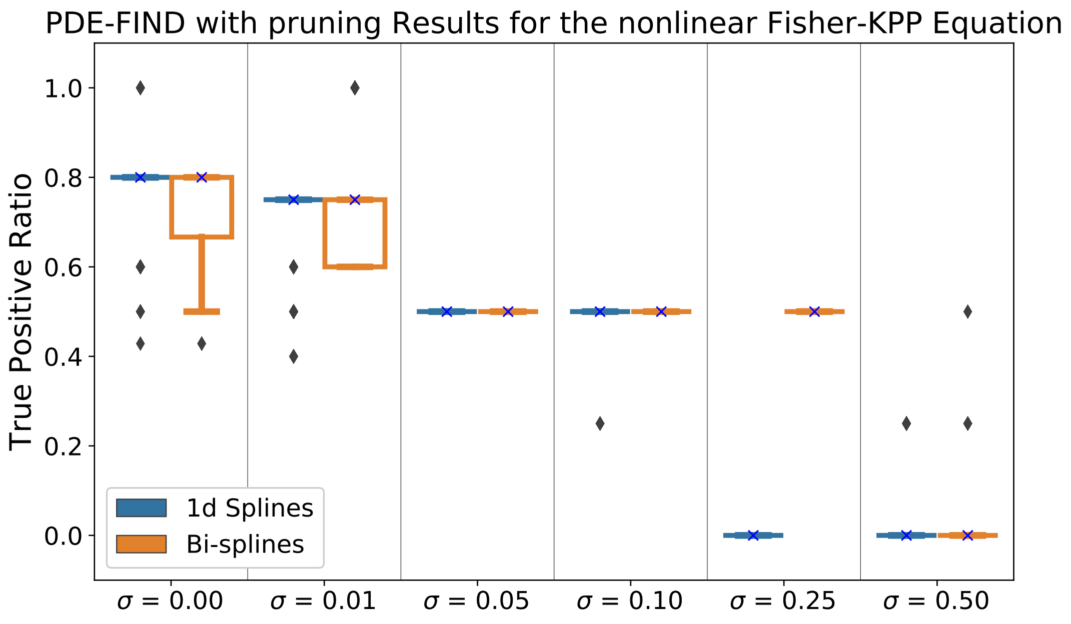

Figure S4: TPR values for the nonlinear Fisher-KPP equation when using 1-dimensional cubic splines versus cubic bi-splines for denoising data and approximating partial derivatives.

on the validation data.

For a given value of s, we find the spline locations and coefficents in a similar manner to that described in [1, § 3.2.6]. We first fit the global spline model to the training data using a constant variance (OLS) error model. This first-pass spline computation allows us to estimate $u(x_i, t_j)$ for the training data and in turn estimate $\mathcal{J}(\theta)$. We now iteratively fit the cubic spline method to the training data using $\mathcal{J}(\theta)$ with $\gamma = 1.0$ until the number of knot locations (determined by the Scipy **bisplev** algorithm) does not change by more than one over five consecutive calculations. We then fix the knot locations and iteratively estimate the spline coefficients, $c_{k,l}$, by minimizing $\mathcal{J}(\theta)$ with $\gamma = 1$ until either the maximum of the relative absolute difference between consecutive estimates, *i.e.*, the inf-norm, converges within 10^{-2} or 100 such computations have been performed.

S4 PDE-FIND without pruning results

We found that using PDE-FIND without pruning results in learning the wrong equation when applied to data from biological transport models, even when no noise is added to the data. We evaluated accuracy, using the true positive ratio (TPR) as a metric, for the diffusion-advection (Figure S5), Fisher-KPP (Figure S6), and nonlinear Fisher-KPP equations (Figure S7).

For the diffusion-advection equation, we found that the TPR value of the final learned equation when using ANN approximations is higher for all values of σ when using pruning with PDE-FIND than without pruning (Figure S5). In general, for small values of σ, we observed that pruning enables PDE-FIND to better learn the true equation when using CV local spline and finite difference computations, but it harms the ability to learn the true equation for larger values of σ. For example, the median TPR value increases after pruning when using finite difference approximations from TPR = 0.33 to 0.5 for $\sigma = 0$. However, the TPR instead decreases from TPR = 0.33 to 0 at $\sigma = 0.05$ and from TPR = 0.5 to 0 at $\sigma = 0.10$. The median TPR value when using spline approximations increases from TPR = 0.3 to 0.5 at $\sigma = 0$ and from TPR = 0.33 to 1 at $\sigma = 0.01$. At $\sigma = 0.10$, the median values decreased from TPR = 1.0 to 0.5.

For the Fisher-KPP equation, the median TPR value for PDE-FIND with the ANN computations always increases after using pruning (Figure S6). The median value for PDE-FIND with finite difference computations increases for $\sigma = 0, 0.01$, but decreases from TPR = 0.5 to 0 for $\sigma = 0.05$ and from TPR = 0.45 to 0 for $\sigma = 0.10$. The median TPR value for PDE-FIND with CV local spline computations increases for $\sigma = 0, 0.01, 0.05$, and 0.10, but decreases from TPR = 0.4 to 0 at both $\sigma = 0.25$ and 0.50. Thus, pruning always helped PDE-FIND learn the true equation when using the ANN method, and helps the other computational methods for small noise levels.

For the nonlinear Fisher-KPP qquation, the median TPR value always improved the accuracy of the PDE-FIND method when using ANN approximations (Figure S7). When using finite difference approximations, the median TPR value increases for $\sigma = 0, 0.01.05$. The median value decreases from TPR = 0.3 to 0 at $\sigma = 0.10$ for finite difference approximations. When using CV local spline approximations, the median TPR value increases when $\sigma = 0$ and 0.01. The median TPR value decreased from TPR = 0.3 to 0 at $\sigma = 0.50$. While the median value is never TPR = 1 for this equation, these results suggest that pruning in general helps reduce the number of incorrect terms in the library.

S5 Tables of learned PDEs

This section contains tables of the final learned PDEs for data from each equation considered at a given noise level. The equation form is the one most commonly selected by the PDE-FIND

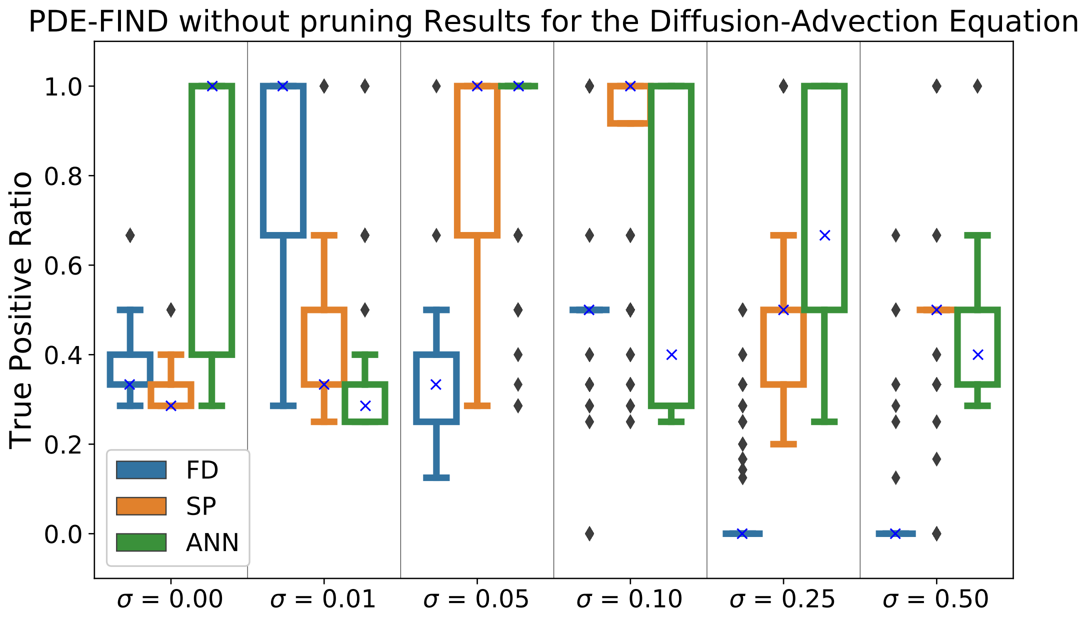

Figure S5: TPR values for the diffusion-advection equation.

method with pruning over the 1,000 different training-validation splits of u_t and Θ. The provided parameter values are the mean value for these parameters when the equation form was the final learned equation.

References

[1] H. Banks, W. C. Thompson, and S. Hu. *Modeling and Inverse Problems in the Presence of Uncertainty*. CRC Press, Boca Raton, 2014.

[2] P. Dierckx. An Algorithm for Surface-Fitting with Spline Functions. *IMA J Numer Anal*, 1(3):267–283, July 1981.

[3] J. T. Nardini and D. M. Bortz. The influence of numerical error on parameter estimation and uncertainty quantification for advective PDE models. *Inverse Problems*, 35(6):065003, June 2019.

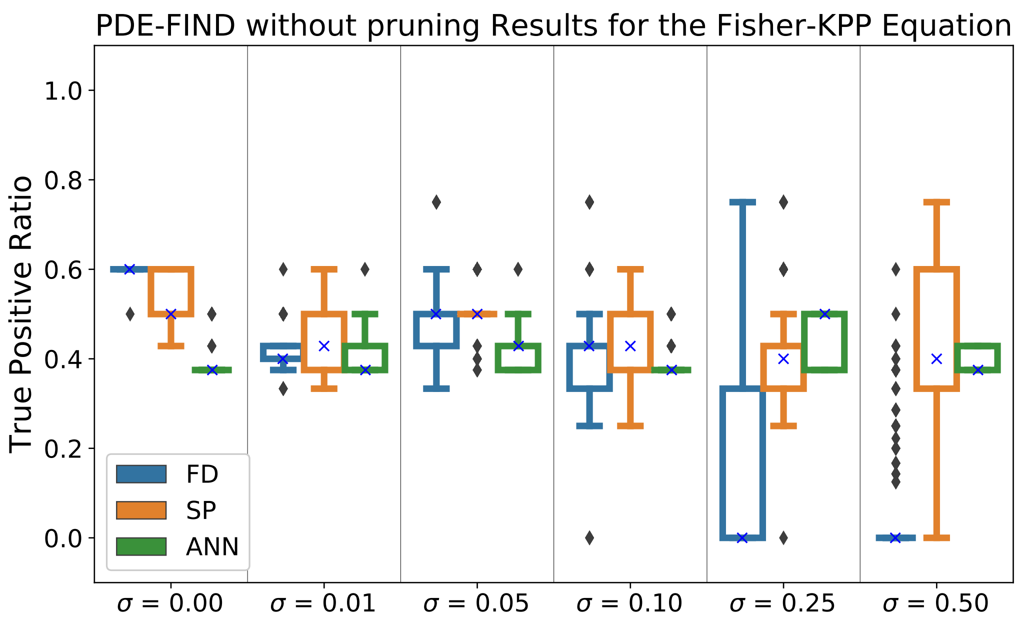

Figure S6: TPR values for the Fisher-KPP equation.

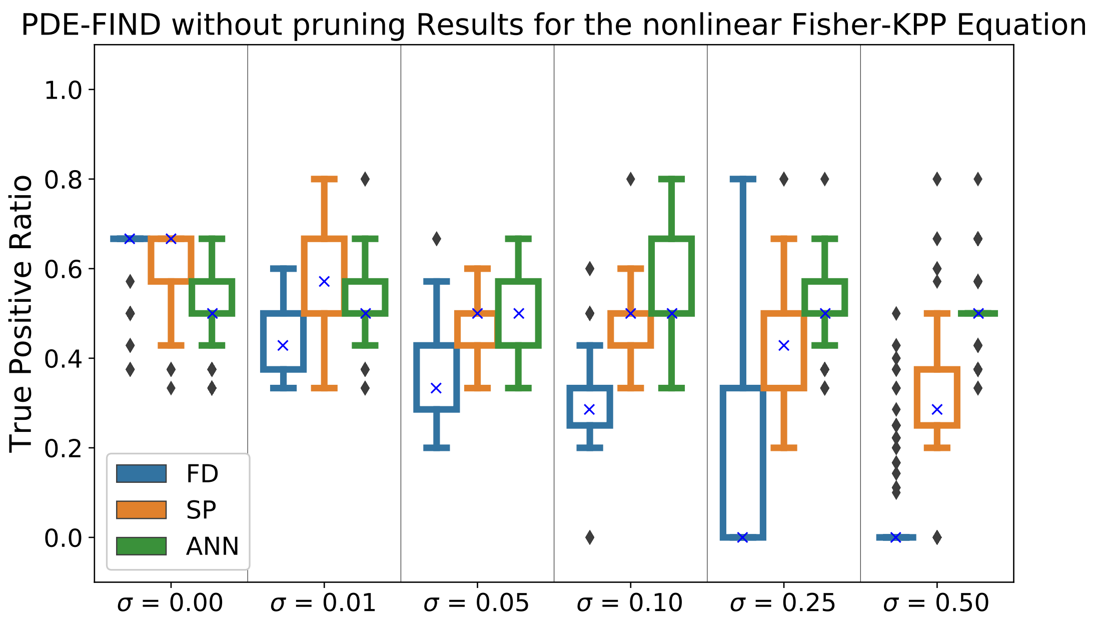

Figure S7: TPR values for the nonlinear Fisher-KPP equation.

		True Equation
		$u_t = 0.01u_{xx} - 0.8u_x$
$\boldsymbol{\sigma}$	**Method**	**Learned Equation**
0.0	FD	$u_t = -0.799974u_x + 0.010414u_{xx} - 0.000350u^2u_x$
0.01	FD	$u_t = -0.800226u_x + 0.009940u_{xx}$
0.05	FD	$u_t = 0$
0.10	FD	$u_t = 0$
0.25	FD	$u_t = 0$
0.50	FD	$u_t = 0$
0.0	LCVSP	$u_t = -0.807744u_x + 0.011464u_{xx} + 0.000670u^2u_x + 0.000012u^2u_{xx}$
0.01	LCVSP	$u_t = -0.793993u_x + 0.011877u_{xx}$
0.05	LCVSP	$u_t = -0.796512u_x + 0.012290u_{xx}$
0.10	LCVSP	$u_t = -0.774238u_x$
0.25	LCVSP	$u_t = -0.709333u_x$
0.50	LCVSP	$u_t = 0$
0.0	LNCVSP	$u_t = -0.820792u_x + 0.011752u_{xx}$
0.01	LNCVSP	$u_t = -0.819847u_x + 0.011726u_{xx}$
0.05	LNCVSP	$u_t = -0.819094u_x + 0.012085u_{xx}$
0.10	LNCVSP	$u_t = -0.790680u_x$
0.25	LNCVSP	$u_t = -0.760747u_x$
0.50	LNCVSP	$u_t = -0.729489u_x$
0.0	GNCVSP	$u_t = -0.764276u_x$
0.01	GNCVSP	$u_t = -0.694783u_x$
0.05	GNCVSP	$u_t = -0.611367u_x$
0.10	GNCVSP	$u_t = -0.581202u_x$
0.25	GNCVSP	$u_t = 0$
0.50	GNCVSP	$u_t = 0$
0.0	ANN	$u_t = -0.809223u_x + 0.010963u_{xx}$
0.01	ANN	$u_t = -0.802903u_x + 0.011107u_{xx} - 0.000074u^2u_{xx} + 0.000765u_x^2$
0.05	ANN	$u_t = -0.810360u_x + 0.010693u_{xx}$
0.10	ANN	$u_t = -0.808996u_x + 0.009535u_{xx}$
0.25	ANN	$u_t = -0.795770u_x + 0.009386u_{xx}$
0.50	ANN	$u_t = -0.801987u_x + 0.007888u_{xx} + 0.000835u_x^2$

Table S1: Learned Equations for the diffusion-advection equation.

		True Equation
		$u_t = 0.02u_{xx} + 10.0u - 10.0u^2$
$\boldsymbol{\sigma}$	**Method**	**Learned Equation**
0.0	FD	$u_t = 0.020086u_{xx} - 9.994321u^2 + 9.995583u$
0.01	FD	$u_t = -10.154641u^2 + 9.950923u$
0.05	FD	$u_t = 0$
0.10	FD	$u_t = 0$
0.25	FD	$u_t = 0$
0.50	FD	$u_t = 0$
0.0	LCVSP	$u_t = 0.020474u_{xx} - 9.993088u^2 + 9.996574u$
0.01	LCVSP	$u_t = 0.019600u_{xx} - 9.972429u^2 + 9.977920u$
0.05	LCVSP	$u_t = -10.130441u^2 + 9.925709u$
0.10	LCVSP	$u_t = -10.087935u^2 + 9.916230u$
0.25	LCVSP	$u_t = 0$
0.50	LCVSP	$u_t = 0$
0.0	LNCVSP	$u_t = 0.020435u_{xx} - 9.991312u^2 + 9.994767u$
0.01	LNCVSP	$u_t = 0.019522u_{xx} - 9.977385u^2 + 9.982515u$
0.05	LNCVSP	$u_t = -10.121782u^2 + 9.916090u$
0.10	LNCVSP	$u_t = -10.087677u^2 + 9.926292u$
0.25	LNCVSP	$u_t = 0$
0.50	LNCVSP	$u_t = 0$
0.0	GNCVSP	$u_t = -10.264500u^2 + 10.229065u$
0.01	GNCVSP	$u_t = -8.598153u^2 + 9.375428u$
0.05	GNCVSP	$u_t = -10.346971u^2 + 10.122676u$
0.10	GNCVSP	$u_t = -10.007866u^2 + 10.075741u$
0.25	GNCVSP	$u_t = -9.312518u^2 + 9.304600u$
0.50	GNCVSP	$u_t = -5.621682u^2 + 7.104374u$
0.0	ANN	$u_t = 0.023272u_{xx} - 9.307794u^2 + 9.533177u$
0.01	ANN	$u_t = 0.023017u_{xx} - 9.397175u^2 + 9.600546u$
0.05	ANN	$u_t = 0.020534u_{xx} - 9.733768u^2 + 9.837442u$
0.10	ANN	$u_t = 0.022343u_{xx} - 9.287166u^2 + 9.587605u$
0.25	ANN	$u_t = 0.011912u_{xx} - 11.160631u^2 + 12.537031u$ $+0.071219uu_{xx} - 0.105350u_x^2$
0.50	ANN	$u_t = -0.015750u_x + 0.013682u_{xx} - 8.688903u^2$ $+12.179728u - 0.034142u^2u_x + 0.077472uu_{xx} - 0.109284u_x^2$

Table S2: Discovered Equations for the Fisher-KPP Equation

		True Equation
		$u_t = 0.020000uu_{xx} + 0.020000u_x^2 + 10.000000u - 10.000000u^2$
$\boldsymbol{\sigma}$	**Method**	**Learned Equation**
0.0	FD	$u_t = 0.000178u_{xx} - 9.996233u^2 + 9.996516u + 0.019671uu_{xx} + 0.019729u_x^2$
0.01	FD	$u_t = -10.268294u^2 + 10.210714u$
0.05	FD	$u_t = -9.531702u^2 + 9.731417u$
0.10	FD	$u_t = 0$
0.25	FD	$u_t = 0$
0.50	FD	$u_t = 0$
0.0	LCVSP	$u_t = 0.000760u_{xx} - 10.013375u^2 + 10.013144u + 0.018798uu_{xx} + 0.018324u_x^2$
0.01	LCVSP	$u_t = 0.005632u_{xx} - 9.820662u^2 + 9.789693u + 0.017619u_x^2$
0.05	LCVSP	$u_t = -10.393255u^2 + 10.287537u$
0.10	LCVSP	$u_t = -10.264929u^2 + 10.194679u$
0.25	LCVSP	$u_t = -10.146329u^2 + 10.082739u$
0.50	LCVSP	$u_t = 0$
0.0	LNCVSP	$u_t = 0.000738u_{xx} - 10.011882u^2 + 10.011901u + 0.018874uu_{xx} + 0.018394u_x^2$
0.01	LNCVSP	$u_t = 0.005585u_{xx} - 9.828360u^2 + 9.796878u + 0.017437u_x^2$
0.05	LNCVSP	$u_t = -10.396293u^2 + 10.285593u$
0.10	LNCVSP	$u_t = -10.333053u^2 + 10.259315u$
0.25	LNCVSP	$u_t = -9.875062u^2 + 9.794690u$
0.50	LNCVSP	$u_t = 0$
0.0	GNCVSP	$u_t = -0.025827u_{xx} - 10.407031u^2 + 10.439599u$
0.01	GNCVSP	$u_t = -0.010336u_{xx} - 10.663675u^2 + 10.575536u$
0.05	GNCVSP	$u_t = -0.009571u_{xx} - 10.670700u^2 + 10.563083u$
0.10	GNCVSP	$u_t = -9.792682u^2 + 9.824648u$
0.25	GNCVSP	$u_t = -0.017669u_{xx} - 10.233510u^2 + 10.177806u$
0.50	GNCVSP	$u_t = -0.023890u_{xx} - 9.392645u^2 + 9.576821u$
0.0	ANN	$u_t = 0.009869u_{xx} - 9.295491u^2 + 9.237594u - 0.032329uu_{xx} + 0.016924u_x^2$
0.01	ANN	$u_t = -9.398164u^2 + 9.389853u + 0.024833u_x^2$
0.05	ANN	$u_t = 0.009080u_{xx} - 9.311857u^2 + 9.245655u - 0.032236uu_{xx} + 0.016673u_x^2$
0.10	ANN	$u_t = -0.006456u_{xx} - 8.965042u^2 + 9.140129u + 0.027012u_x^2$
0.25	ANN	$u_t = -0.010203u_{xx} - 7.927777u^2 + 8.551556u + 0.034169u_x^2$
0.50	ANN	$u_t = 0.285757 - 0.026084u_{xx} - 5.419017u^2 + 6.724382u + 0.044730u_x^2$

Table S3: Discovered Equations for the nonlinear Fisher-KPP Equation.